UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

SISTEMAS INFORMÁTICOS

Desarrollo de un sistema web para el análisis de datos en bioinformática utilizando técnicas de factorización positiva de matrices

Autores: José Luís García Sarasa
Gloria Pérez de la Varga

Profesor Director: Alberto Pascual Montano

Curso académico 2006/2007

ÍNDICE

INDICE DE FIGURAS

AGRADECIMIENTOS

A nuestros padres por transmitirnos mediante su ejemplo lo que es la disciplina y la fuerza de voluntad en el trabajo y habernos ayudado a alcanzar las metas que nos hemos propuesto.

A nuestros compañeros y amigos por su apoyo y ayuda incondicional.

Y finalmente, pero no por ello menos importante, a Silvia y Gerardo, por haber compartido con nosotros todo el largo camino que nos ha llevado hasta aquí, por su paciencia, cariño y comprensión.

AUTORIZACIÓN

José Luís García Sarasa y Gloria Pérez de la Varga, como autores de este proyecto, autorizan a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado".

Firmado:

Jose Luis García Sarasa

Gloria Pérez de la Varga

....

RESUMEN

RESUMEN

Este proyecto tiene como fundamento la integración de diversas tecnologías para construir una aplicación Web capaz de satisfacer las necesidades y requerimientos actuales en el campo del análisis de datos biológicos.

La funcionalidad de la web será similar a la de la herramienta existente *BioNMF*, pero con las ventajas principales de que será accesible desde cualquier ordenador con acceso a la red Internet, sin necesidad de instalar ningún archivo ejecutable, y la mejora significativa en el tiempo de la obtención del resultado, gracias al procesamiento distribuido del análisis de los datos.

Palabras clave: NMF, factorización no negativa de matrices, aplicación web, bioinformática, microchips de ADN, aplicación distribuida.

ABSTRACT

The aim of this project is to integrate diverse technologies to build a Web application able to satisfy the present necessities and requirements in the field of bioinformatics data analysis.

The functionality of the web will be similar to the one of the existing tool *BioNMF*, but with the advantages that it will be accessible from any computer with access to the Internet, with no need to install a exe file, and the significant improvement in the time for obtaining the result, thanks to the distributed processing of the data analysis, distributed application.

Key words: NMF, non negative matrix factorization, Web application, bioinformatics, DNA microchips.

INTRODUCCIÓN

El presente documento recoge la explicación y documentación del proyecto "Desarrollo de un sistema web para el análisis de datos bioinformáticos utilizando técnicas de factorización no negativa de matrices". En él se da una detallada explicación de la funcionalidad de la herramienta con ejemplos gráficos y capturas de pantalla, así como una completa documentación sobre el desarrollo y análisis llevado a cabo para la implementación del mismo.

Se comienza con una pequeña explicación de las tecnologías empleadas y el uso que de cada una de ellas se ha hecho en el proyecto.

Una vez introducido el objeto de este proyecto, se procede a comentar más concretamente cada una de las fases que se producen en el tratamiento de la información, desde su búsqueda, transformación y hasta su representación.

Se detallan las ventajas del empleo de este sistema web sobre las herramientas ya existentes, así como los posibles inconvenientes.

Se finaliza con una serie de ideas sobre posibles actualizaciones o mejoras sobre el sistema web, que podrían formar parte de un trabajo futuro.

MOTIVACIÓN

En los últimos años la técnica de factorización no negativa de matrices ó NMF desarrollada en 1999 por Lee y Seung [1] se ha convertido en una herramienta muy popular para el análisis e interpretación de datos y muy especialmente para el análisis de datos relacionados con la bioinformática, debido a su capacidad para proporcionar información relevante sobre las relaciones entre conjuntos experimentales de datos.

En este campo en particular, el algoritmo de NMF se ha venido utilizando en contextos tales como el análisis de expresión génica [2], análisis de secuencias de proteínas [3], caracterización funcional de genes [4] e incluso el análisis de documentación científica [5].

La principal diferencia entre NMF y otras técnicas clásicas de factorización reside en la restricción a entradas no negativas. Esto lleva a una representación basada en partes de los datos, ya que solo se permiten combinaciones aditivas de los mismos. De esta forma los factores producidos por este método se pueden considerar como subconjuntos de elementos que tienden a las mismas características en subporciones del conjunto de datos.

Por otro lado, los métodos clásicos descomponen los datos de la matriz en conjuntos nuevos de matrices de cualquier signo, lo que implica cancelaciones muy complejas de elementos positivos y negativos para reconstruir el conjunto de datos original. Por tanto, que el método NMF tenga unas propiedades más sencillas que hacen que la descomposición de la matriz generada por el algoritmo se entienda, en contraposición a los métodos clásicos y que proporcione resultados de formas más intuitivas que los otros métodos, entre otras, han hecho que la comunidad científica, y en particular en el campo de la bioinformática, haya incrementado su interés en los últimos años en este método.

Pero aunque el algoritmo es capaz de ofrecer datos de gran relevancia dada su enorme capacidad de producir resultados fácilmente interpretables, es también cierto que el mismo presenta una complejidad computacional elevada y además requiere de una presentación e interactividad adecuada para que la interpretación de los resultados que produce sea efectiva. Por ello la mayoría de las implementaciones solo están disponibles en MATLAB, en programas de líneas de comandos o integradas en paquetes de análisis mas extensos, y aunque estas implementaciones funcionan correctamente, a veces es necesaria una herramienta estándar, simple e intuitiva para llevar a cabo algunos tipos de análisis mas específicos en un entorno integrado y que no implique al investigador tener conocimiento amplio del uso de la herramienta en la que se encuentre implementado.

Esta situación ha motivado que se lleve a cabo el sistema web para el análisis de datos bioinformáticas como una aplicación estándar, versátil, sencilla de usar, que no requiere instalación de ningún tipo y que puede usarse para muchas de las aplicaciones del campo de la bioinformática.

NMF: NON NEGATIVE MATRIX FACTORIZATION

NMF es un algoritmo de factorización de matrices propuesto originalmente por Lee y Seung para el análisis de imágenes faciales. Esta técnica se puede aplicar al análisis exploratorio de datos como método de proyección para reducir la dimensionalidad de los datos o para descubrir patrones ocultos, aunque su aplicación más extendida es para facilitar la interpretación de los datos [10].

Mediante NMF podemos reproducir de forma aproximada una matriz de datos, $V \in R^{n \times m}$, como un producto de dos matrices $W \in R^{n \times k}$ y $H \in R^{k \times m}$. En este contexto, W representa un conjunto reducido de k factores (*vectores base*), mientras que H contiene los coeficientes de la combinación lineal de factores necesaria para la reconstrucción de V. A este conjunto de coeficientes se les denomina *vectores codificantes*.

Adicionalmente, se suelen imponer las siguientes restricciones:

- k<<n.
- V, W y H no tienen valores negativos.
- Las columnas de W están normalizadas (la suma de los elementos de cada columna vale 1).

Una de las aplicaciones que tiene este algoritmo en bioinformática es el *Análisis de expresión de genes*, donde V representa una matriz con la expresión de n genes en m experimentos. En este caso, las k columnas de W se les denomina experimentos base, mientras que cada fila de H representa un gen base. Podemos ver su correspondencia en la Figura 1:

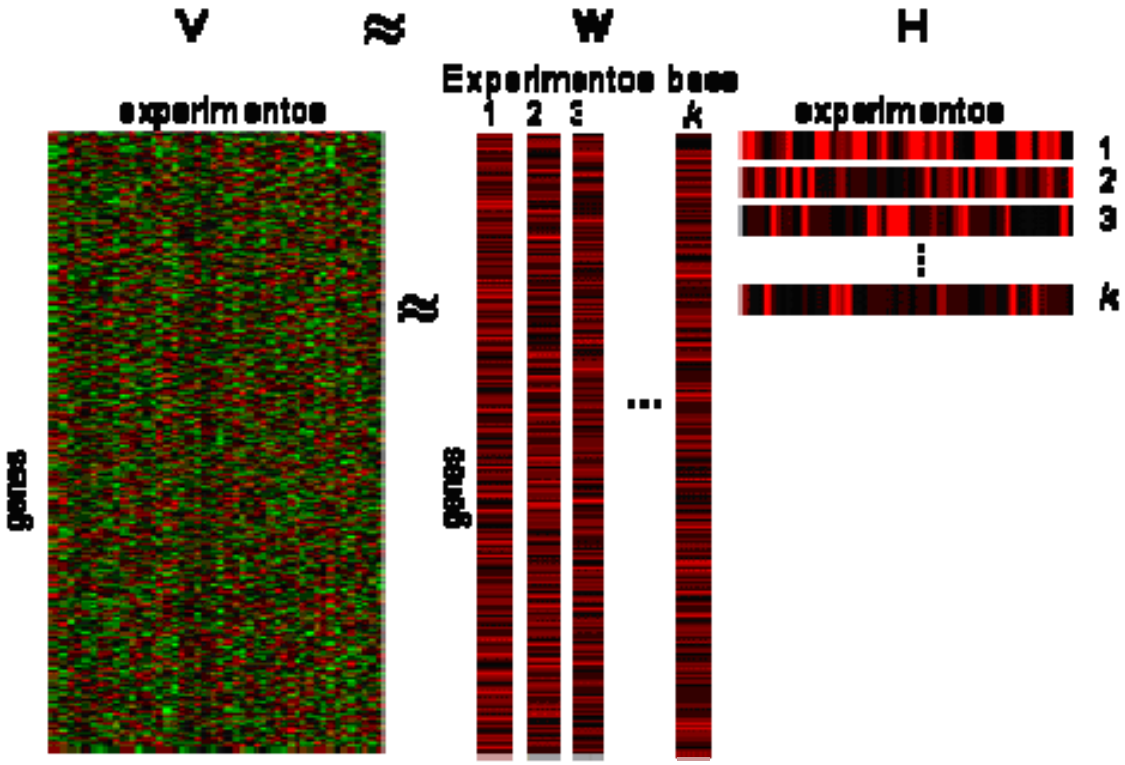

Figura 1: Modelo de expresión génica

La mayor diferencia entre NMF y otras técnicas de factorización que han sido aplicadas al análisis de expresión de genes, tales como *Análisis de Componentes Principales (PCA), Descomposición de Valores Singulares (SVD) o Análisis de Componentes Independientes (ICA)* [6-8], radica en la restricción impuesta a los vectores base (W) y a los codificantes (H) de no utilizar valores negativos. Gracias a esta restricción, se obtiene una representación de los datos

basada en partes o secciones, ya que sólo se permiten combinaciones aditivas, nunca sustractivas.

De esta manera, los factores pueden ser interpretados como partes de los datos o, dicho de otro modo, como elementos que tienden a aparecer juntos. Por el contrario, otras técnicas de factorización como las mencionadas anteriormente, permiten que los valores de W y H tengan cualquier signo, lo que conlleva a cancelaciones complejas de elementos positivos y negativos durante la reconstrucción del conjunto original de datos. En otras palabras, NMF tiende a producir factores que permiten una interpretación contextual relativamente sencilla, mientras que aquellos obtenidos mediante las otras técnicas no contienen un "significado" intuitivo.

OBJETIVOS

El objetivo principal de este proyecto es diseñar y desarrollar un sistema web que permita el uso sencillo y remoto de flujos de análisis de datos en bioinformática, utilizando el algoritmo de NMF y algunas de sus variantes.

Hoy en día los investigadores que usan este tipo de herramientas quieren tener una visión rápida y aproximada de los resultados, que se utilice escasos recursos y que se pueda usar independientemente de la plataforma software en la que trabajen, solamente contado con un acceso a Internet.

Con la herramienta web que se desarrolla en este proyecto se pretenden conseguir todos estos objetivos, gracias a que se trata de un sistema web y que la parte de las operaciones se ejecuta en un sistema distribuido en computadores de la Universidad Complutense, proporcionando la rapidez de consecución de resultados pedida.

Además nos planteamos otros objetivos secundarios enfocados ha hacer mas amigable el uso y trabajo al investigador que utilice la herramienta, y son que el sistema web acepte ficheros de entrada con varios formatos (con o sin etiquetas de filas y columnas), posibilidad de aviso en el correo electrónico, disponibilidad de los resultados en el servidor durante un tiempo determinado, posibilidad de guardar las imágenes generadas como parte de los resultados, …

CONTENIDO DE LA MEMORIA

La memoria esta estructurada de la siguiente forma:

- Después de la introducción se procede a un pequeño repaso de las tecnologías y herramientas empleadas en el proyecto.

- La Historia del Proyecto nos lleva por los pasos que se han dado para desarrollar el proyecto y comprender las distintas tecnologías empleadas.

- Conclusiones y Trabajos futuros. En este punto se valida si se han alcanzado los objetivos iniciales, posibles ampliaciones y comentarios adicionales sobre la realización del proyecto.

- Apéndices.

- Por último se incluye la bibliografía usada como apoyo a lo largo del desarrollo del proyecto.

TÉCNICAS Y HERRAMIENTAS UTILIZADAS

METODOLOGÍA: PROCESO UNIFICADO DE DESARROLLO

El Proceso Unificado de Desarrollo es un framework o marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones. Estas iteraciones ofrecen como resultado un *incremento* del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.
Cada una de estas iteraciones se divide a su vez en una serie de pasos: Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc…

El Proceso Unificado está centrado en la arquitectura; es decir, se asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura software de un sistema.

Una parte muy importante a tener en cuenta en el proceso unificado son los riesgos. Se deben identificar los riesgos en críticos en una fase temprana del ciclo de desarrollo. Los resultados de cada iteración se deben seleccionar en el orden que asegure que los riesgos principales se traten primero.

ECLIPSE

El Proyecto Eclipse es un proyecto de desarrollo de software de código abierto dedicado a proporcionar una plataforma industrial robusta, con amplias características y con calidad comercial para el desarrollo de herramientas altamente integradas.

Está compuesto de tres subproyectos: la Plataforma Eclipse, la Java Development Tool y el Plug-in Development Environment. El éxito de la Plataforma Eclipse viene de cómo es capaz de admitir una amplia gama de

herramientas de desarrollo para reproducir lo mejor posible las herramientas existentes en la actualidad.

Pese a que Eclipse esté escrito en su mayor parte en Java (salvo el núcleo), se ejecute sobre máquina virtual de ésta y su uso más popular sea como un IDE para Java, Eclipse es neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, etc.

La característica clave de Eclipse es la extensibilidad. Eclipse es una gran estructura formada por un núcleo y muchos plug-ins que van conformando la funcionalidad final. La forma en que los plug-ins interactúan es mediante interfaces o *puntos de extensión*; así, las nuevas aportaciones se integran sin dificultad ni conflictos.

En cuanto a las aplicaciones clientes, eclipse provee al programador con frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, …

EL LENGUAJE DE PROGRAMACIÓN: JAVA

JAVA es un lenguaje de programación orientado a objetos que fue desarrollado por Sun Microsystems a principio de los noventa. [12]

El lenguaje toma mucha de su sintaxis de C y de C++, pero su modelo de objetos es más simple y elimina algunas de las herramientas de bajo nivel, como los punteros.

JAVA es independiente de la plataforma, es decir, permite la ejecución de un mismo programa en múltiples sistemas operativos.

JAVA puede funcionar como una aplicación sola o como un "applet", que es un pequeño programa hecho en Java. Los applets de Java se pueden "pegar" a una página HTML en un servidor web, y con esto se obtiene un programa que cualquiera que tenga un navegador compatible podrá usar.

La programación en Java permite el desarrollo de aplicaciones bajo el esquema de *Cliente Servidor* como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente, y de esta forma logra distribuir el trabajo a realizar.

El JRE (Java Runtime Environment) es el software necesario para ejecutar cualquier aplicación desarrollada para la plataforma Java. El usuario final usa el JRE como parte de paquetes software o plugins (o conectores) en un navegador Web. Sun ofrece también el SDK de Java 2, o JDK (Java Development Kit) en cuyo seno reside el JRE, e incluye herramientas como el compilador de Java, Javadoc para generar documentación o el depurador. Puede también obtenerse como un paquete independiente, y puede considerarse como el entorno necesario para

ejecutar una aplicación Java, mientras que un desarrollador debe además contar con otras facilidades que ofrece el JDK [12].

Struts

Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC (Modelo Vista Controlador) bajo la plataforma J2EE (Java 2, Enterprise Edition).

Struts implementa un solo controlador (ActionServlet) que evalúa las peticiones del usuario mediante un archivo configurable (struts-config.xml). [17]

Los componentes del modelo corresponden a la lógica del negocio con la cual se comunica la aplicación web. Usualmente el modelo comprende accesos a bases de datos o sistemas que funcionan independientemente de la aplicación web.

Los componentes de control son los encargados de coordinar las actividades de la aplicación, que van desde la recepción de datos del usuario, las verificaciones de forma y la selección de un componente del modelo a ser llamado. Por su parte los componentes del modelo envían al control sus eventuales resultados o errores de manera de poder continuar con otros pasos de la aplicación.

Esta separación simplifica enormemente la escritura tanto de vistas como de componentes del modelo: Las páginas JSP no tienen que incluir manejo de errores, mientras que los elementos del control simplemente deciden sobre el paso siguiente. [14,16]

Entre las características principales de Struts podemos destacar:

- Configuración del control de forma centralizada.
- Las interrelaciones entre acciones se especifican por medio de instrucciones XML en vez de codificarlas directamente en los programas.
- Incluye librerías de entidades para facilitar la mayoría de las acciones que normalmente realizan las páginas JSP.

Struts permite que el desarrollador se concentre en el diseño de aplicaciones complejas como una serie simple de componentes del Modelo y de la vista intercomunicados por un control centralizado, y obtener así una aplicación más consistente y más fácil de mantenerse.

SERVIDOR DE APLICACIONES

Un servidor de aplicaciones es un dispositivo software que proporciona servicios de aplicación a los ordenadores cliente de forma distribuida. Normalmente es el servidor de aplicaciones el que gestiona la mayoría de las funciones de lógica de negocio y de acceso a datos. Se puede deducir por tanto que aplicando tecnologías de servidores de aplicación se obtienen grandes niveles de centralización y en consecuencia menor dificultad en el desarrollo de aplicaciones (facilita el mantenimiento de la aplicación). Otras características importantes son la alta disponibilidad y la escalabilidad

Entre los servidores de aplicaciones podemos destacar WebLogic, WebSphere u OracleApplicationServer, JBoss, etc… Tipicamente, y debido al éxito del lenguaje JAVA, se conoce por servidor de aplicaciones a una implementación de la especificación J2EE, aunque hay algunos casos en los que no es así.

Los servidores de aplicaciones suelen incluir software de conectividad, APIs, …. Al mismo tiempo que soportan una amplia variedad de estándares, como HTML, SSL, ..

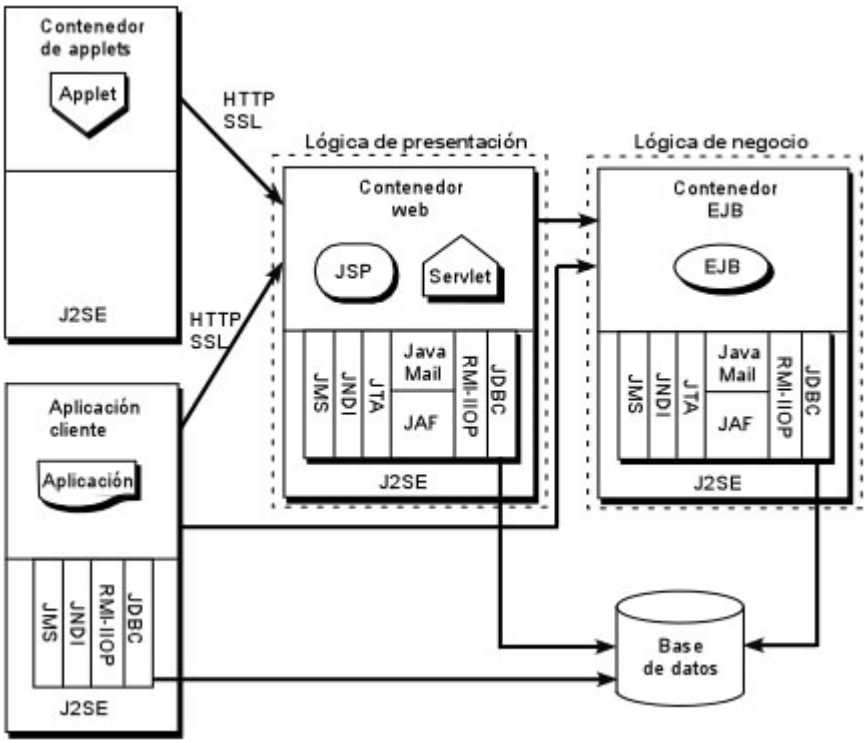

Figura 2: Estructura de un servidor de aplicaciones

Apache Tomcat

Una de las partes que componen la arquitectura J2EE es el *contenedor o servidor web*. Esta es la parte 'visible' del servidor de aplicaciones que utiliza los protocolos HTTP y SSL para comunicarse.

Apache Tomcat es un contenedor de servlets bajo licencia de Apache que implementa de forma 'open source' las especificaciones de los servlets y JSP (JavaServerPages) de Sun Microsystems; es decir, un contenedor de servlets es un shell de ejecución que maneja e invoca servlets por cuenta del usuario. [15,17]

MATLAB

Matlab es a la vez un entorno interactivo y un lenguaje de programación técnica de alto nivel. Como lenguaje nos permite construir herramientas reusables que se pueden agrupar en 'Toolbox', que son conjuntos de archivos que amplían el entorno Matlab y que sirven para trabajar en problemas particulares, como pueden ser los sistemas económicos, la inteligencia artificial, la simulación de sistemas, …

Por su funcionalidad, Matlab permite realizar tareas de cálculo complejas de forma más rápida que con los lenguajes de programación tradicionales, y además Está disponible para las plataformas Unix, Windows y Mac OS X.

De forma general nos permite realizar:

- Desarrollo de algoritmos y aplicaciones.
- Cálculo numérico.
- Análisis y acceso a datos.
- Visualización de datos (funciones gráficas bidimensionales y tridimensionales).
- Publicación de resultados y distribución de aplicaciones.

SHELL DE UNIX

El shell de UNIX es el intérprete de comandos de los sistemas pertenecientes a la familia UNIX, a través del cual interactuamos con el sistema cuando no se emplea un entorno gráfico.

El funcionamiento básico se basa en que el usuario da una línea de comandos y el shell transfiere las órdenes al sistema operativo, que las ejecuta sobre la máquina.

Algunas shells ofrecen características avanzadas, como puede ser la ejecución en segundo plano, alias, …

Algunos ejemplos de shells son bash, ash, csh, …

EMPLEO DE LAS HERRAMIENTAS EN EL PROYECTO

Puesto que es una metodología de trabajo, el Proceso Unificado de Desarrollo lo hemos usado como tal, es decir, como una guía para avanzar a lo largo de todo el camino del proyecto. Hay que decir que dado el reducido número de componentes en el grupo no se han podido aplicar todas sus características plenamente, pero así todo se ha intentado seguir lo máximo posible.

Eclipse como entorno de desarrollo integrado, junto con struts, se han utilizado para el diseño e integración de todos los componentes (interfaz gráfica, fichero de configuración, jsps, etc…) que conforman la aplicación web final.

El servidor de aplicaciones se encargará de albergar todas las páginas JSP desarrolladas [16].

La comprobación de las opciones del usuario y la realización de las operaciones con la matriz de entrada se realiza en un fichero en Matlab que se pasa como parámetro a MATLAB.

Para llamar a MATLAB pasándole el script para que se ejecute el análisis solicitado por el investigador usamos un script de shell de UNIX que se envía al *cron* de UNIX, que es un administrador regular de procesos en segundo plano que ejecuta programas a intervalos regulares. (Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el archivo *crontab*).
Además, para este mismo fin también se creó un **demonio**, que es un tipo especial de proceso que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario (es un proceso no interactivo). Este tipo de programas se ejecutan de forma continua, es decir, que funciona sin tener relación con un terminal o consola.
Este programa realiza la misma funcionalidad que el crontab, pero de esta forma nos despreocupamos de si tendremos permisos suficientes para ejecutar el crontab en caso de que en algún momento el sistema web se aloje en otro servidor.

HISTORIA DEL PROYECTO

ENFOQUE

Desde un principio la idea mas clara respecto al proyecto era la facilidad y comodidad en el manejo por parte del usuario final de la herramienta que se iba a desarrollar. Por tanto ha sido siguiendo esta premisa que se ha enfocado todo el desarrollo del proyecto.

Para ello la apariencia o interfaz debía ser lo mas parecida posible a la del programa existente, pero con el potencial de su uso en la Web, además de la mejora en el tiempo de obtención de la respuesta al cálculo solicitado en cada momento.

Respecto a la organización del proyecto, se ha basado sobre todo en el correo electrónico y en la mensajería instantánea, dada la incompatibilidad de horarios de los miembros del grupo para realizar reuniones físicas. Esta forma de comunicación se ha completado con reuniones directas con el profesor director para tratar los aspectos más relevantes del proyecto, como mostrar prototipos y solucionar problemas concretos.

En un principio pensamos usar repositorios de datos donde alojar el código fuente y la documentación del proyecto, pero debido a que solo éramos dos integrantes observamos que no era necesario.

En el nivel de desarrollo del proyecto se ha intentado desde un primer momento seguir el enfoque del Proceso Unificado de Desarrollo, pero debemos decir que no siempre ha sido posible, por diferentes motivos, y se ha combinado en algunas etapas con el método de la programación extrema.

El planteamiento que nos hicimos para tomar esta decisión fue que el modelo de programación extrema aportaría agilidad a los cambios de requisitos o mejoras de los prototipos, mientras que el modelo incremental permitiría realizar tantas versiones como fuesen necesarias de cada uno de los prototipos para optimizar su funcionamiento antes de ampliarlo con nuevas características.

FUNCIONAMIENTO

Indicaremos a continuación de forma breve el funcionamiento del sistema web desarrollado, con el fin de tener desde este punto una visión global de lo que la herramienta nos puede ofrecer.

Se introduce en el sistema web ficheros de texto con los datos a analizar. Éstos pueden contener etiquetas de cabeceras de filas de columnas o no.

A continuación se procede, si es necesario, a la normalización de los datos, y posteriormente se puede elegir algún método para transformar a positivos los valores negativos que los datos iniciales pudieran tener.

Una vez transformados los datos se pasa al análisis de los mismos. El usuario puede elegir entre tres tipos de análisis:

- *NMF Estándar*.

  Implementa el algoritmo clásico NMF de Lee y Seung.
  Simplemente calcula las matrices H y W en que se descompone la matriz original de datos.

- *Biclustering [2].*

  Implementa una variante del algoritmo NMF conocido como ns-NMF [9].
  A partir de la matriz de entrada agrupa de forma aleatoria filas y columnas para formar clusters o grupos de muestras altamente relacionadas.

- *Sample Clasification].*

  Este método fue propuesto por Brunet y al. en 2004 [19], y trabaja sobre la idea de reducir la dimensionalidad de los datos de entrada. Para ello el algoritmo NMF basado en esta técnica agrupa las muestras en clusters.
  Debido a la componente aleatoria de las matrices de entrada este método propone 'medir' la estabilidad del conjunto de muestras, y para ello calcula la matriz consensus de la original (para cada par de muestras calcula la probabilidad de que pertenezcan al mismo cluster) y el coeficiente de correlación cofenético (establece una medida de la estabilidad del subconjunto, tomando valores entre 0 y 1, que indican la menor y la mayor estabilidad del subconjunto respectivamente).

En función del tipo de análisis seleccionado se introducen los parámetros adecuados.

En este punto, el usuario lanza el análisis, indicando si quiere que se le envíe un email de aviso cuando esté listo el resultado del análisis. Se asigna un identificador único por consulta que se le notifica y el resto del proceso es transparente para el usuario, y que se detalla más en profundidad en los siguientes apartados.

ANÁLISIS Y DISEÑO

Objetivos

- Repartir la funcionalidad del sistema en una estructura que facilite la comprensión, modificación, mantenimiento y reutilización del software final.
- Estudiar ventajas e inconvenientes de las posibles tecnologías a usar.
- Crear un punto de partida para la implementación.

Desarrollo

El análisis de requisitos de nuestro programa se llevó a cabo básicamente tomando la especificación directa de requisitos por parte del profesor. De esta forma también se obtuvieron los casos de uso, ya que el profesor director nos indicó claramente el uso concreto que se haría del programa en todos los casos posibles.

Se completó la obtención de requisitos por medios de la observación del programa en el que está basado nuestro proyecto (Bionmf) y su uso, ya que la funcionalidad, al menos desde el punto de vista del usuario final debía ser la misma, siendo los posibles cambios que se efectuaran a la aplicación transparentes para el mismo.

De la información obtenida en la toma de requerimientos y del análisis de los casos de uso, obtuvimos el modelo global del sistema web, empleado en el desarrollo del mismo, y representado en el siguiente diagrama:

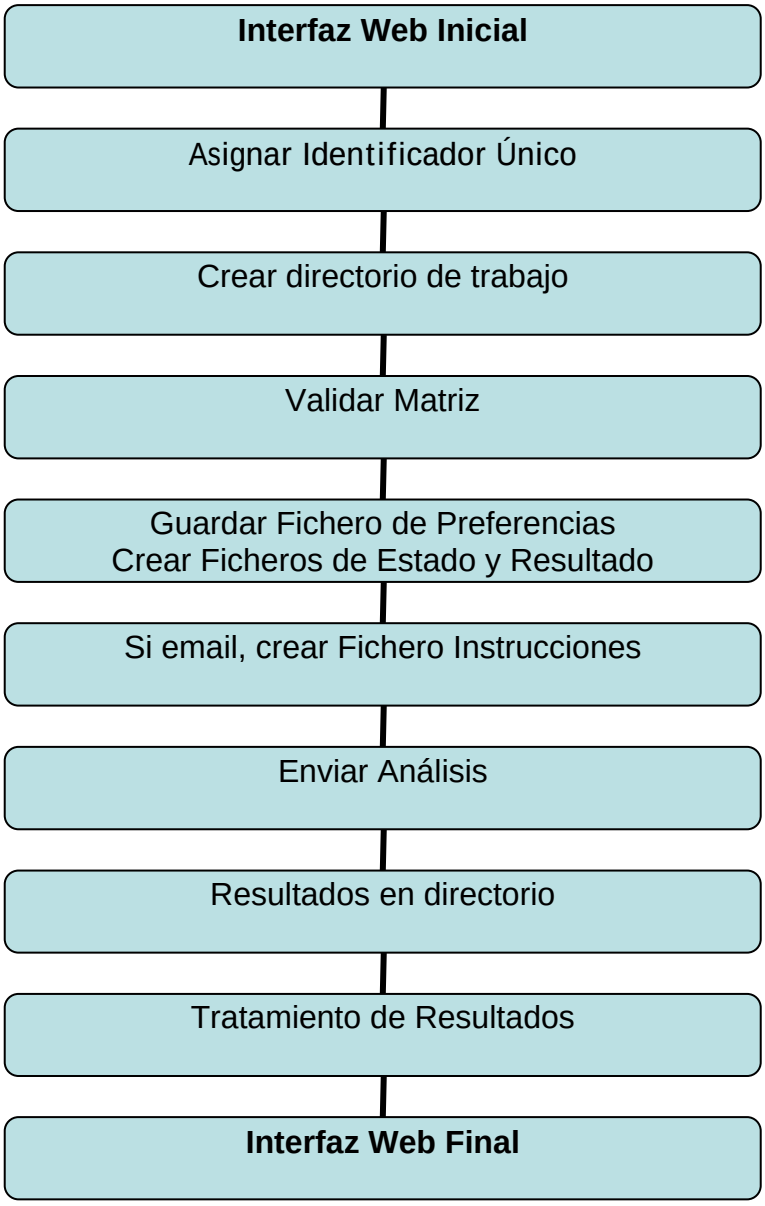

Figura 3: Modelo global de la aplicación Web

IMPLEMENTACIÓN

Objetivos

- Implementar el sistema en término de componentes: ficheros de código fuente, scripts,…
- Planificar las integraciones de sistema necesarias en cada momento.
- Implementar los subsistemas definidos durante el diseño.

Desarrollo

Hemos seguido un modelo de implementación incremental, ya que de esta forma se puede obtener una versión ejecutable del sistema muy pronto, facilita la localización de defectos durante las pruebas y la integración y las pruebas correspondientes, al trabajar sobre una parte pequeña, son mas minuciosas.

Ahora indicaremos de forma más detallada cada parte o módulo que compone el sistema web, siguiendo el modelo global obtenido en la fase de análisis y diseño.

Interfaz gráfica para introducción de los datos

Esta parte de la interfaz gráfica es la página web principal del sistema y el punto de entrada a la misma. Actualmente se accede a la misma tecleando en el navegador:

http://bisaurin.dacya.ucm.es:8090/bioNMF/

Desde esta página el investigador introduce tanto el fichero con los datos de entrada como las opciones que requiera para el análisis de los mismos. También en este punto indica, si quiere, su email, para recibir por este método una notificación cuando el resultado del análisis que haya pedido esté listo.

La apariencia de esta página es la siguiente:

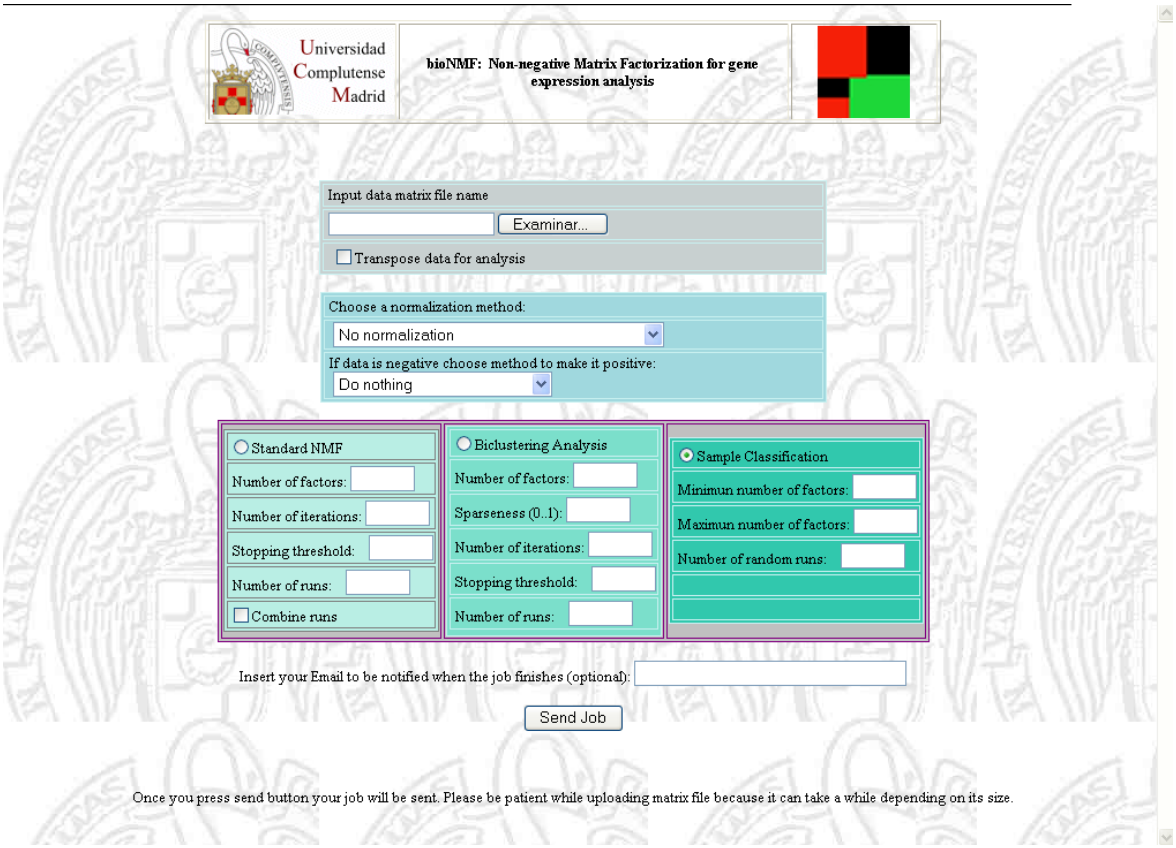

Figura 4: Página web principal

A nivel de implementación esta página se corresponde con el fichero i*ndex.jsp*.

Una vez que el usuario envía la petición es el módulo de struts en función de su configuración el que decide los siguientes pasos a realizar. De forma general, el flujo de información que se seguiría desde este punto viene representado en el siguiente modelo:

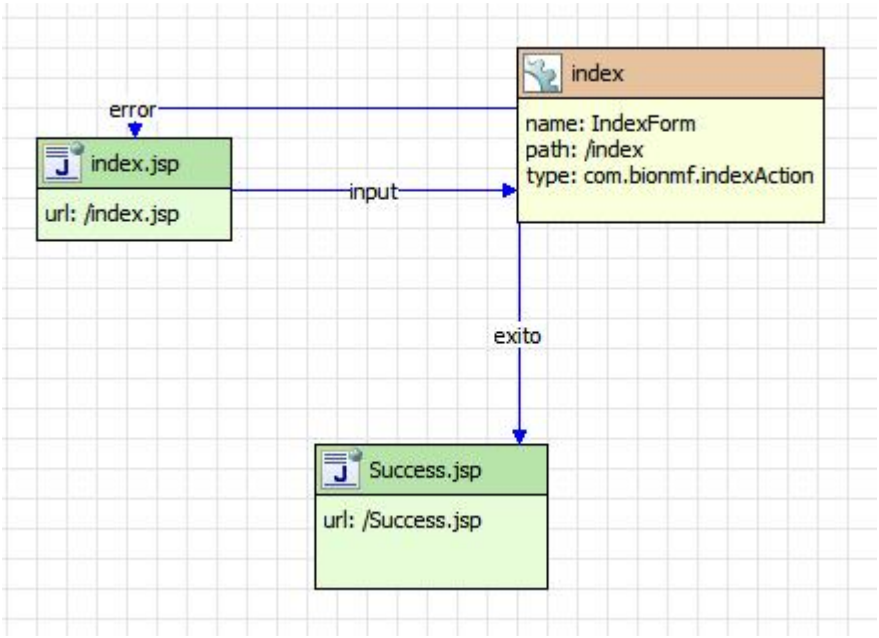

Figura 5: Diagrama de Struts

El fichero de struts '*indexAction*' es el que guía la acción, comprobando si se produce algún error y en ese supuesto redirige a la página inicial, o si todo está correcto nos lleva a la página representada por el fichero *Success.jsp*.

Validaciones

Una vez que el usuario ha introducido los datos y ha enviado la petición, *indexAction.java* ejecuta la clase *Engine.java*.
Esta clase de encarga de:

- Asignar un identificador único a la petición.
- Crear el directorio de trabajo asociado a ese identificador único en la jerarquía de carpetas del sistema web.
- Subir la matriz a ese directorio.
- Validar la matriz de entrada.

En el caso de que en cualquier punto de este proceso se produjese un error, se borra el directorio de trabajo con todo su contenido y la aplicación nos devuelve a la página inicial informándonos del error encontrado.

Por tanto las clases [11,13] que interviene en esta parte de la aplicación son:

Clase	Tarea
indexAction.java	Guía la acción
Engine.java	Ejecuta tareas validación
GeneradorUUID.java	Genera identificador único
MatrixValidator.java	Valida la matriz

La validación de la matriz con los datos de entrada consiste a grandes rasgos en comprobar si la matriz es numérica; si lo es, la renombra para tratarla en Matlab y la copia en el directorio (MatrixMat.txt), y si no lo es la parte, creando dos ficheros: uno con las cabeceras y otro con los datos, que igualmente copia en el directorio de trabajo (Matrixcab.txt y MatrixMat.txt).

Trabajo con las opciones introducidas por el usuario

Como se puede ver en el diagrama que representa el modelo global del sistema web, una vez que el usuario envía su petición, y siempre que se realicen con éxito las validaciones y operaciones indicadas en el apartado anterior, *indexAction.java* realiza las siguientes tareas, siempre asociadas al directorio de trabajo creado:

- Guarda en un fichero las preferencias del usuario (Preferentes.txt)
- Crea el fichero de estado. (State.txt)
- Crea la página de resultado. (result.jsp)
- Si el usuario ha introducido un email, se crea el fichero de instrucciones para el usuario para la consulta del resultado. (Instructions.txt)
- Copia el script de Matlab (trataprefs.m) de la carpeta de scripts de la aplicación al directorio de trabajo indicado por el identificador único.

Al igual que en la fase de validación, si en cualquiera de estos pasos se produjese un error, se borraría el directorio de trabajo con todo su contenido.

Si por el contrario todas las tareas se llevan a cabo satisfactoriamente, i*ndexAction.java* nos redirige a la página *success.jsp.*

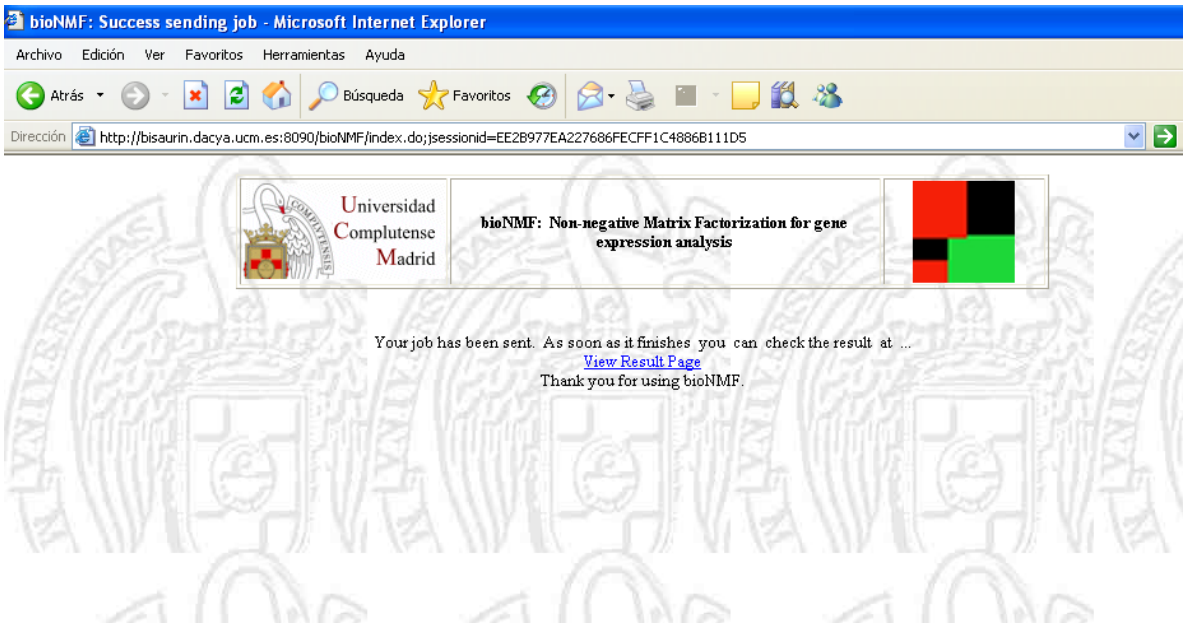

Figura 6: success.jsp

Fichero de estado (state.txt)

Dada la importancia del fichero de estado en la aplicación como motor para la comunicación entre los diferentes procesos, le dedicamos un punto aparte.

El diagrama de estados que le representa es el siguiente, donde se ven claramente los estados que incluye y las acciones que provocan las transiciones que llevan de uno a otro:

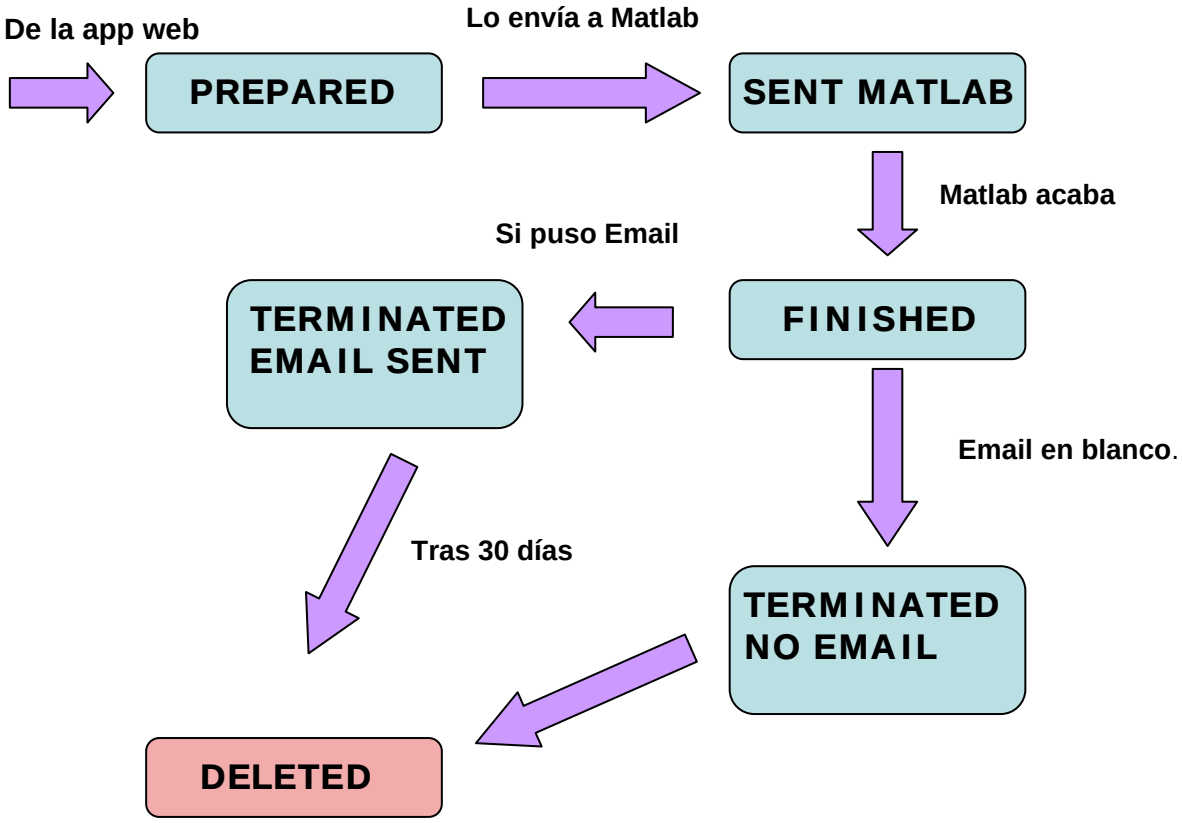

Figura 7: Diagrama de estados

Cuando el fichero es creado por la clase *indexAction.java*, se introduce el estado 'Preparado' como estado inicial.

Este fichero es consultado y modificado por el scanner, que comentamos en el siguiente apartado, en función de las acciones que se vayan produciendo y sus resultados.

Script Shell (scanner)

El script shell de UNIX que usamos, o fichero *scanner* en nuestra aplicación, o bien se envía al *cron* de UNIX cada cierto tiempo o bien lo arranca el programa demonio creado y permanece en memoria continuamente y chequea todos los directorios de trabajo que estén en el sistema de directorios del sistema web en cada momento, permitiendo así que el scanner transite por sus estados de forma adecuada.

Llama a MATLAB pasándole como parámetro el fichero *trataprefs.m* asociado a cada petición y que contiene las instrucciones de MATLAB creadas en función de las opciones seleccionadas por el usuario.

Este fichero también se encarga de comprobar si ha pasado más de un mes desde la fecha en que se realizó una cierta petición, y en ese caso borra el directorio de trabajo asociado.

El scanner se ejecutará de forma distribuida en el servidor remoto de Bisaurin de forma cíclica hasta obtener los resultados esperados.

Mediante este fichero se producen las transiciones en el modelo de estados de la aplicación, actualizando el valor del estado actual en el fichero de estado asociado al directorio de trabajo de la petición.

Fichero de Instrucciones (instructions.txt)

Si el investigador introduce un email se crea un fichero de texto en la clase *indexAction.java* que se le envía en el correo electrónico con las instrucciones que debe seguir para consultar el resultado del análisis solicitado.

Como muestra el diagrama de estados, es el scanner el que chequea si se ha creado este fichero y en caso afirmativo también manda el correo electrónico a la dirección introducida en la pantalla inicial.

Script Matlab (trataprefs.m)

En la carpeta *scripts* de la aplicación reside el fichero de Matlab (trataprefs.m) que se copia directorio de trabajo indicado por el identificador único de cada petición.

trataprefs.m llama al fichero en el que se han recogido las opciones de análisis introducidas por el usuario (*Preferentes.txt*) para leer las opciones seleccionadas (método de transposición, método de normalización, variables del método de normalización elegido,..).

Una vez que tiene las opciones del análisis de la petición lee la matriz de datos ya sin etiquetas, suponiendo que la original las incluyera (*MatrixMat.txt*)

Con las opciones y la matriz de entrada puede obtener ya las instrucciones finales de MATLAB que se ejecutarán en el servidor remoto de Bisaurin, con el fin de obtener los resultados pedidos por el usuario; es decir, normaliza la matriz, la hace positiva y le aplica el método de factorización seleccionado, con el cual obtiene las imágenes correspondientes al análisis y las matrices **VW** y **VH** que guarda en el directorio de trabajo en los ficheros *MatrixW.txt* y *MatrixH.txt*.

Posteriormente, si la matriz inicial contenía etiquetas, crea los ficheros de etiquetas para las matrices VW y VH (ficheros *MatrixWLBL.txt* y *MatrixHLBL.txt*).

Por último, accede al fichero de estado y cambia el estado actual a '*finalizado*'.

Interfaz gráfica para visualización de resultados

A este respecto se pueden dar dos situaciones distintas, y es que el usuario acceda a la página de resultados cuando aún no se ha generado el resultado de su análisis o bien que si esté generado ya.

Si el usuario intenta ver el resultado pero éste aún no se ha generado, se le muestra la página *result.jsp*:

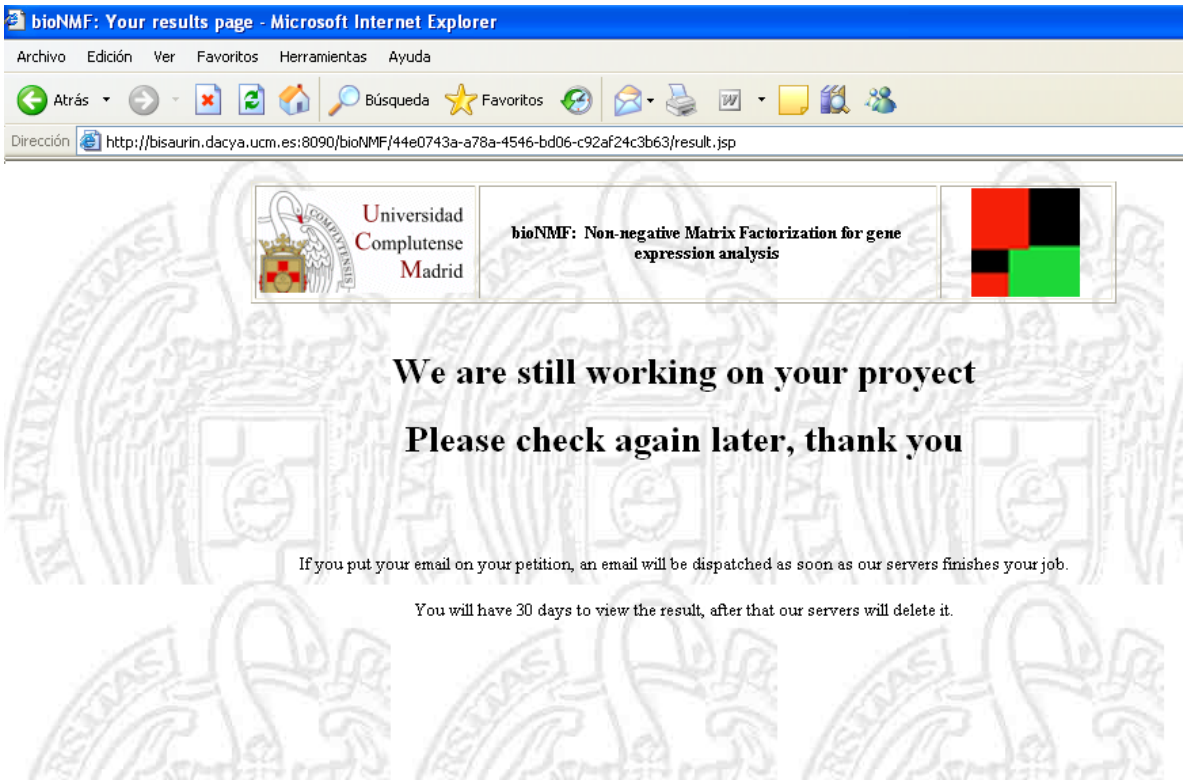

Figura 8: result.jsp

En cambio, si el script de Matlab ya ha generado las imágenes correspondientes al resultado del análisis, (MATLAB finaliza sus operaciones y transita al estado '*finalizado*' la máquina de estados implementada por el fichero *state.txt*), y en función del método de factorización solicitado (estándar, biclustering o simple) el fichero *result.jsp* se sustituirá en el directorio de trabajo por la plantilla correspondiente:

- *resultStandar.jsp*.
- *resultBiclustering.jsp*.

- *resultSample.jsp*.

Posteriormente el scanner tratará el fichero de resultado correspondiente con las variables del fichero de preferencias para que muestre el número de imágenes adecuado (si procede).

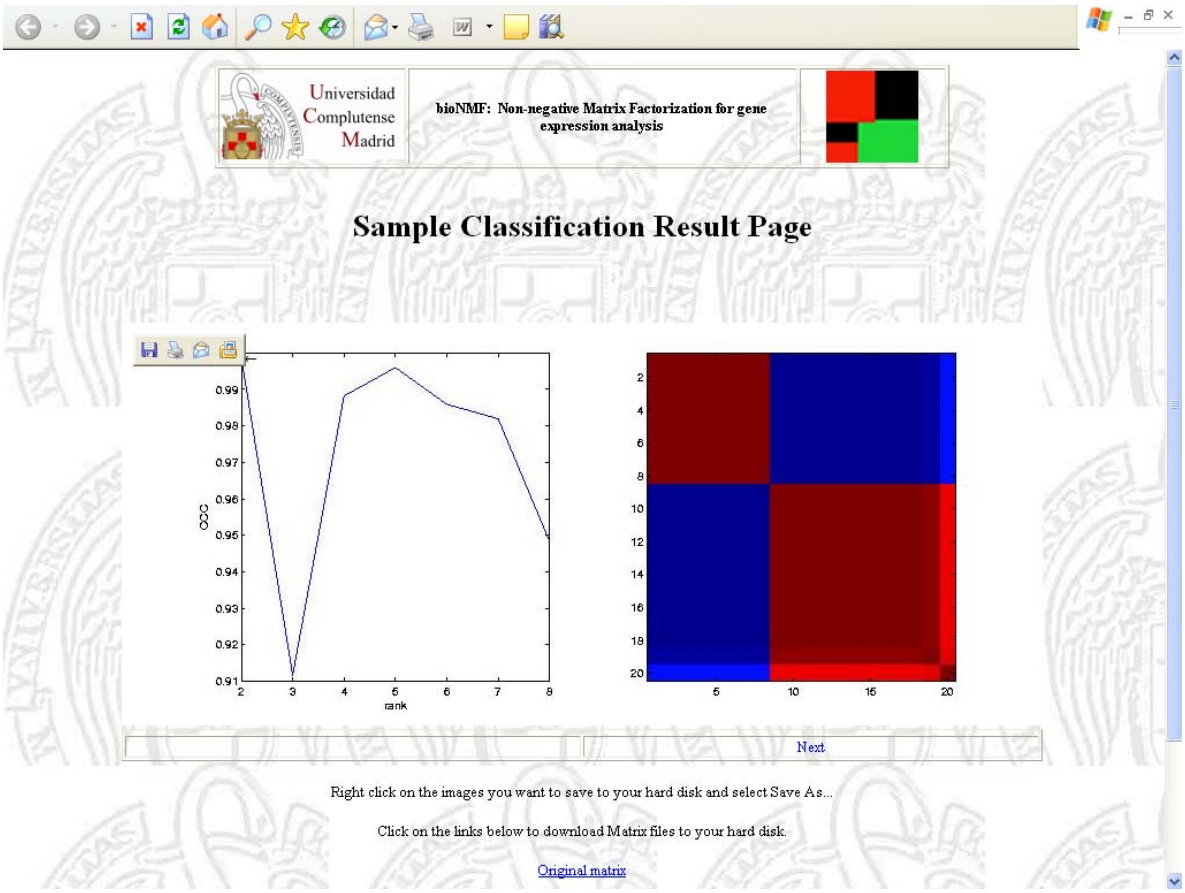

Figura 9: Ejemplo de resultado

PRUEBAS

Objetivos

- Probar los componentes e integrarlos.
- Uso de casos de prueba para encontrar errores aún no descubiertos.

Desarrollo

Las pruebas son una parte fundamental en el desarrollo exitoso de cualquier proyecto. Como tal, hemos ido realizando distintas baterías de pruebas a medida que desarrollábamos cada módulo del proyecto.

Durante todo el desarrollo del proyecto se han realizado tests manuales, al igual que al finalizar la implementación. Estas pruebas se han llevado a cabo tanto por los integrantes del equipo y el profesor, como por posibles usuarios con un total desconocimiento de la herramienta, con el fin de hacer especial hincapié en el aspecto intuitivo del sistema web y la sencillez del mismo.

CASO DE EJEMPLO

1. Cargar el fichero

El sistema web acepta como datos de entrada ficheros de texto separados por comas, que pueden incluir o no etiquetas de fila y/o columna.

Para cargar el archivo presionamos en el botón **Examinar.**

Se puede realizar la transposición de los datos de entrada si éstos están almacenados como filas de datos. Para ello, marcamos **Transpose data for analysis**.

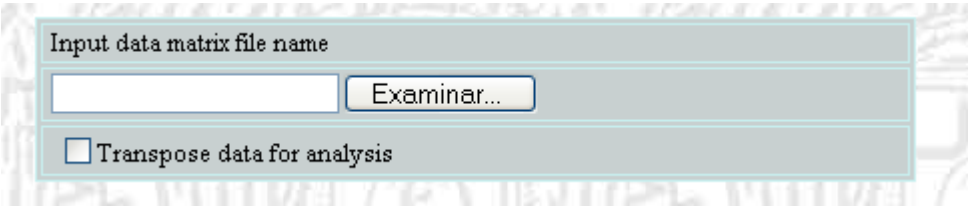

Figura 10: Cargar fichero de datos

2. Transformación de los datos

Normalmente es necesaria una normalización previa de los datos para hacer más evidentes los patrones de los datos de interés. Se incluyen varios métodos de normalización, algunos centran los datos, otros que estandarizan las columnas, etc.…

Se elige el método de normalización en el primer desplegable.

Después de la normalización, y si los datos de entrada contiene valores negativos, se pueden elegir varios métodos para convertirlos en positivos en el segundo desplegable.

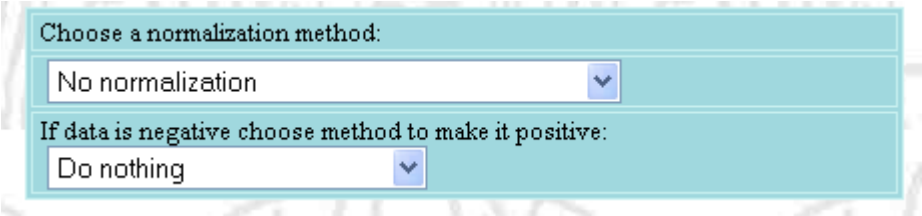

Figura 11: Transformación de los datos

3. Elección del tipo de análisis

En el sistema web implementado se pueden realizar tres tipos de análisis:

- NMF Estandar: el algoritmo clásico NMF de Lee y Seung.
- Biclustering: usa una variante del algoritmo NMF [2,9].
- Sample Clasification: implementa un método de clasificación que usa NMF para clasificar conjuntos experimentales [19].

En función del tipo análisis que queramos realizar tendremos que seleccionar en la pantalla inicial del sistema web unos parámetros u otros [10], como se puede ver en la *figura 12*.

- *Número de factores:* es el rango de la factorización. Se corresponde con el valor *K* del modelo NMF.
- *Número de iteraciones:* número máximo de iteraciones del algoritmo.
- *Umbral de parada:* el algoritmo para cuando los cambios entre las iteraciones de W y H son menores que este umbral.
- *Número de ejecuciones:* número de veces que se repite el algoritmo usando valores iniciales aleatorios de W y H. Marcando la casilla *Combine runs* se combinan todos los resultados obtenidos en un único fichero.

Además de los anteriores, para biclustering [2] tenemos:

- *Sparseness(0...1):* nivel de dispersión. Si es 0 el sistema realiza el algoritmo clásico de NMF, y mientras más se aproxime a 1 el nivel de factorización será menor.

Y para sample clasification [19]:

- *Mínimo y Máximo número de factores:* el rango de la factorización.
- *Número de ejecuciones aleatorias:* número de ejecuciones del algoritmo NMF para calcular el coeficiente de correlación cofenético.

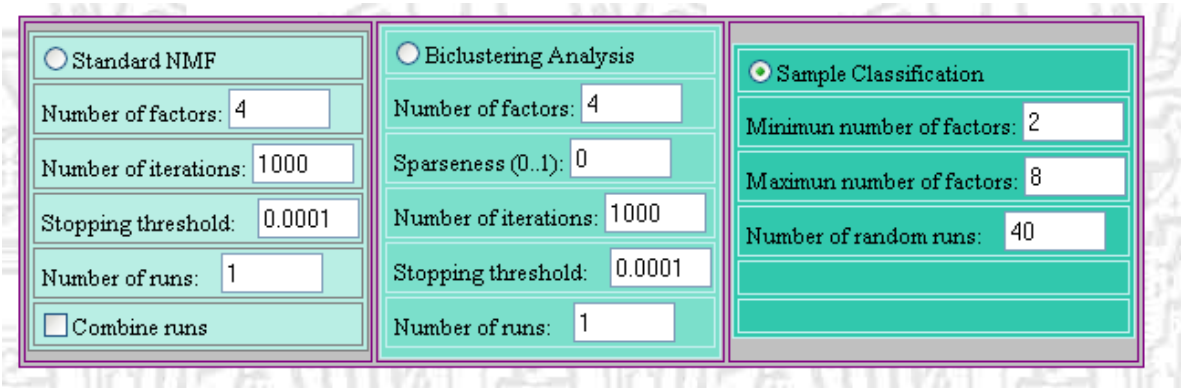

Figura 12: Tipos de análisis

4. Ejecución

Una vez seleccionadas todas las opciones de ejecución ya se puede enviar el trabajo a ejecución.

Previamente, y como paso opcional, se puede introducir un email al que el sistema web nos enviará una notificación cuando el resultado esté disponible en el directorio de trabajo asignado a la petición.

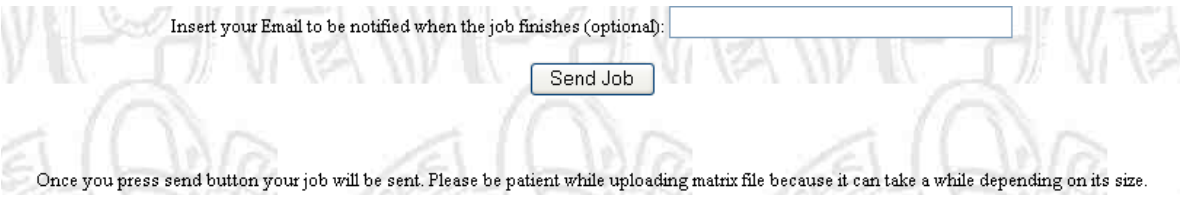

Figura 13: Ejecutar análisis

5. Resultados

En este apartado se muestran ejemplos de la interfaz gráfica de resultados para los distintos tipos de análisis para un mismo fichero de datos de entrada.

Además para los tres tipos de análisis tenemos disponibles como parte del resultado ficheros de texto con la matriz de datos originales y las matrices W y H calculadas, con o sin etiquetas.

Previo a la visualización de los resultados correspondientes al análisis, y siempre que se haya elegido la opción de aviso de obtención de resultados por correo electrónico, el usuario recibirá una notificación en la dirección que haya introducido con el siguiente formato:

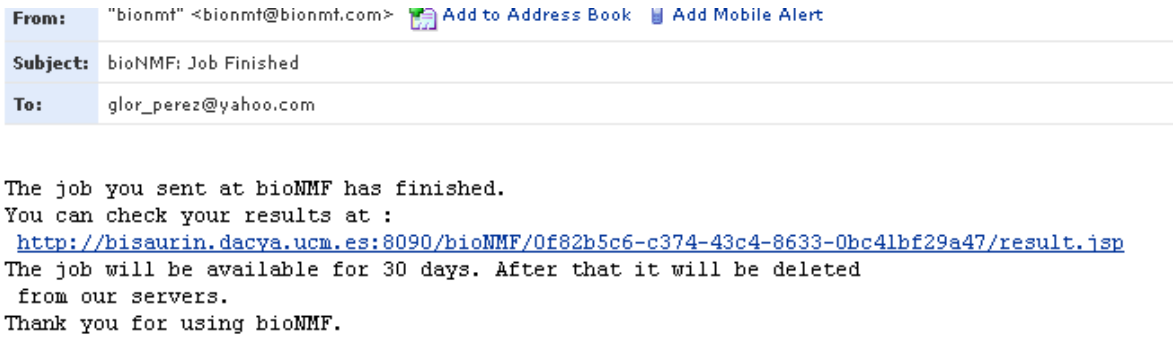

Figura 14: Notificación al correo electrónico

5.1. NMF Estándar.

Este tipo de análisis no muestra ninguna imagen, solamente nos da los ficheros de texto con las distintas matrices que intervienen en la factorización. Así pues, la interfaz gráfica correspondiente a un análisis de este tipo es la siguiente:

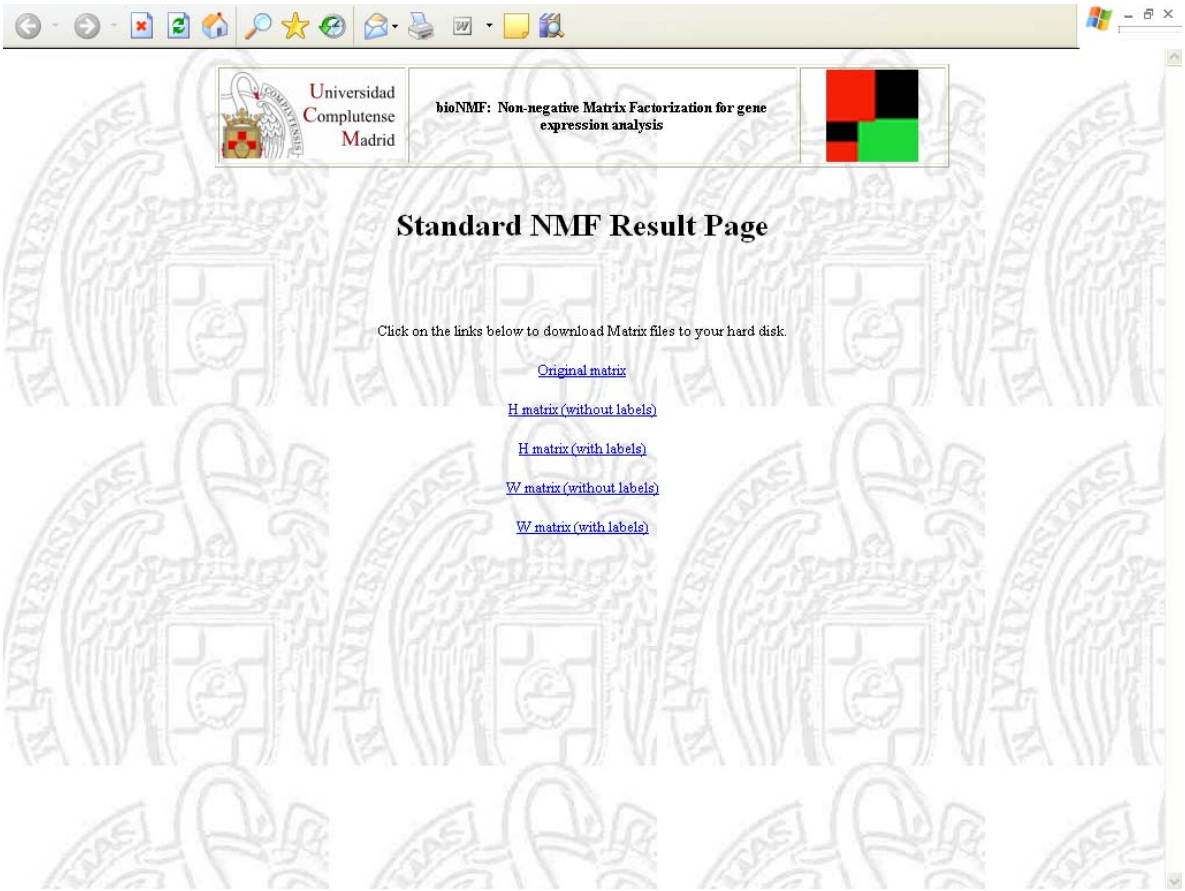

Figura 15: Resultado para NMF estándar

5.2. Biclustering.

La interfaz gráfica para los resultados del biclustering nos ofrecerá los distintos resultados del análisis para cada agrupamiento formado por el cálculo según el número de factores seleccionado en las opciones del análisis.

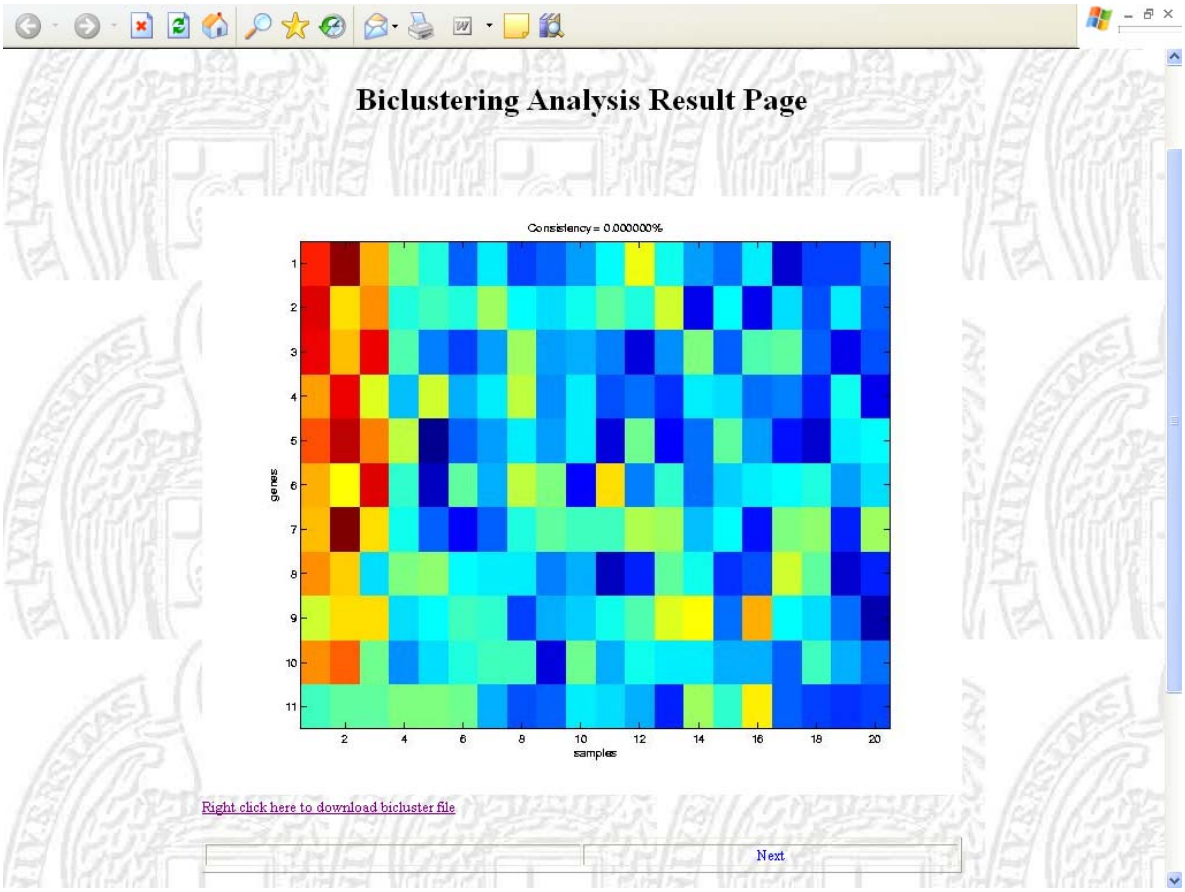

Figura 16: Resultado para Biclustering

Por medio de los botones *Previous/Next* disponibles en la pantalla podremos visualizar cada uno de los clusters o agrupamientos que el algoritmo calcula (tantos como el número de factores seleccionado por el usuario en la pantalla principal). En la figura 17 se muestra una matriz correspondiente a otro de los agrupamientos generados en el mismo análisis:

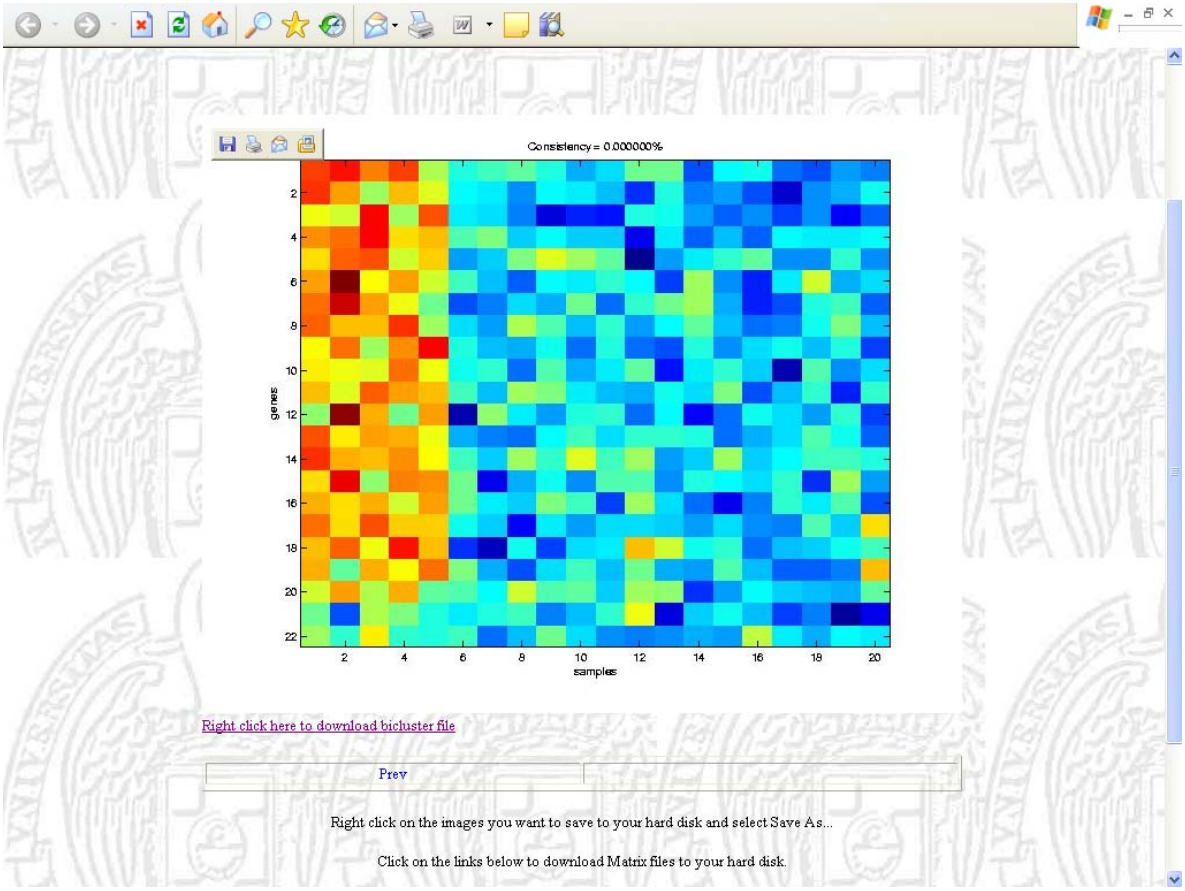

Figura 17: Resultado para Biclustering (II)

Además nos da la posibilidad de guardar el fichero generado por el biclustering y las imágenes generadas. De esta forma el investigador puede tenerlas disponibles para su consulta o uso aunque haya pasado el tiempo predeterminado que se guardan los resultados en el servidor Bisaurin.

Como hemos comentado anteriormente también nos permite ver o guardar ficheros de texto con las matrices generadas y originales. Estos ficheros los abre directamente en el navegador, pero se pueden guardar posteriormente con el formato deseado:

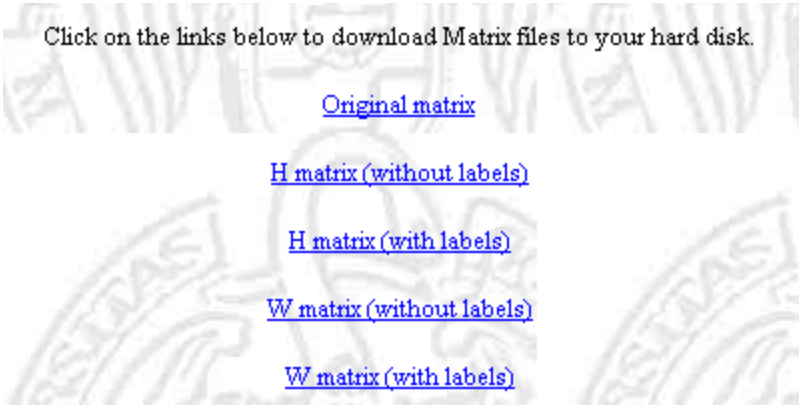

Figura 18: Ficheros resultados

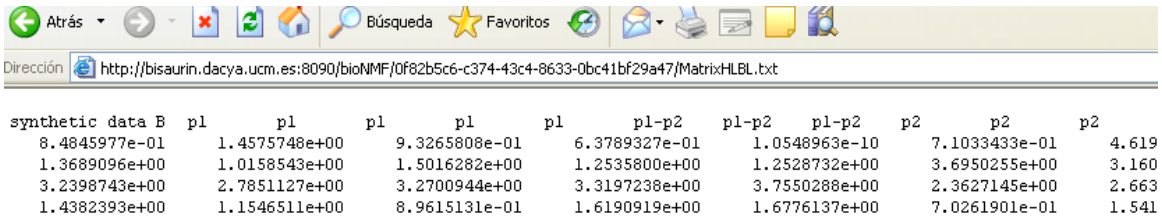

Figura 19: Fichero de texto matriz H

5.3. Sample Clasification.

La interfaz gráfica del resultado muestra la matriz consensus reordenada y el coeficiente de correlación cofenético calculado para cada fila usada en el análisis [19], es decir para el número de factores que se hayan seleccionado (desde MinFactor a MaxFactor):

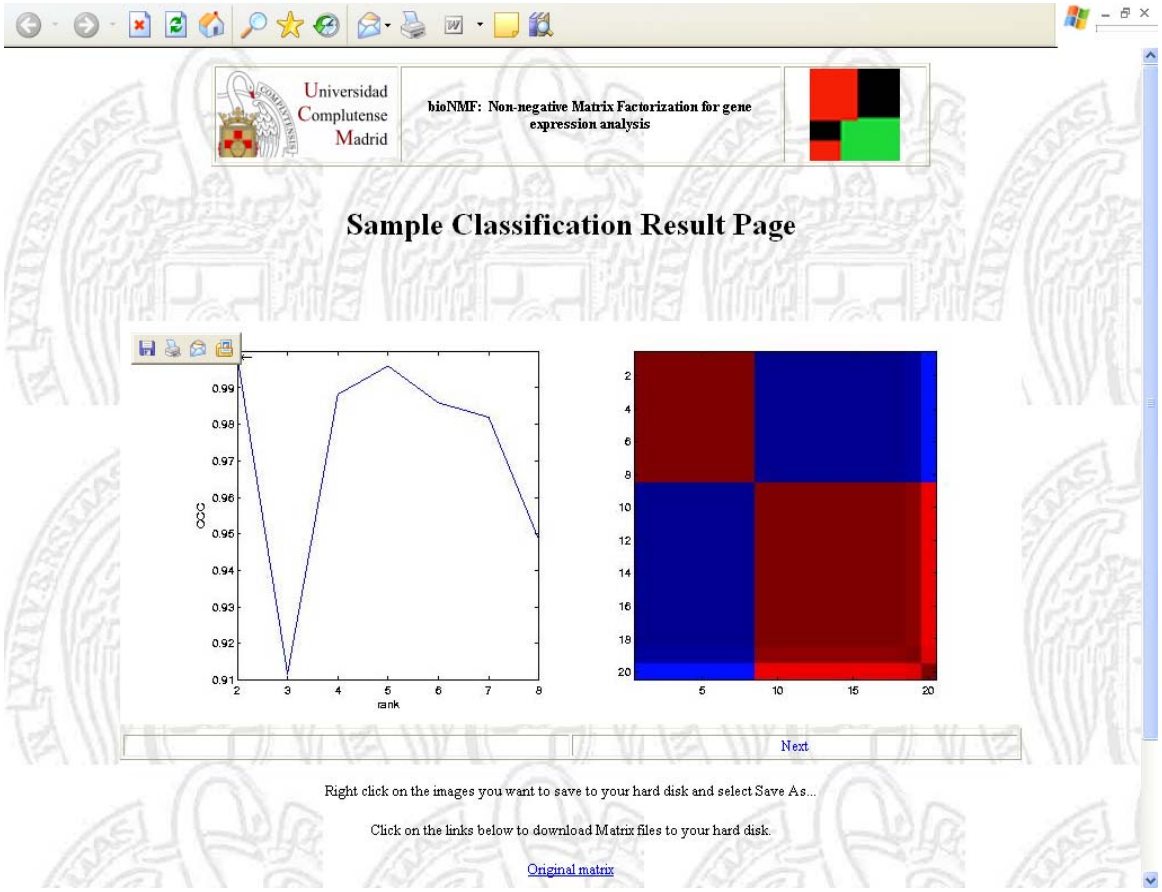

Figura 20: Resultado para Sample Clasification

Por medio de los botones *Previous/Next* se pueden visualizar cada matriz consensus para cada fila y el punto correspondiente del coeficiente cofenético.

La figura 21 muestra la matriz correspondiente al siguiente coeficiente cofenético de los calculados (haciendo clic en el botón *Next*).

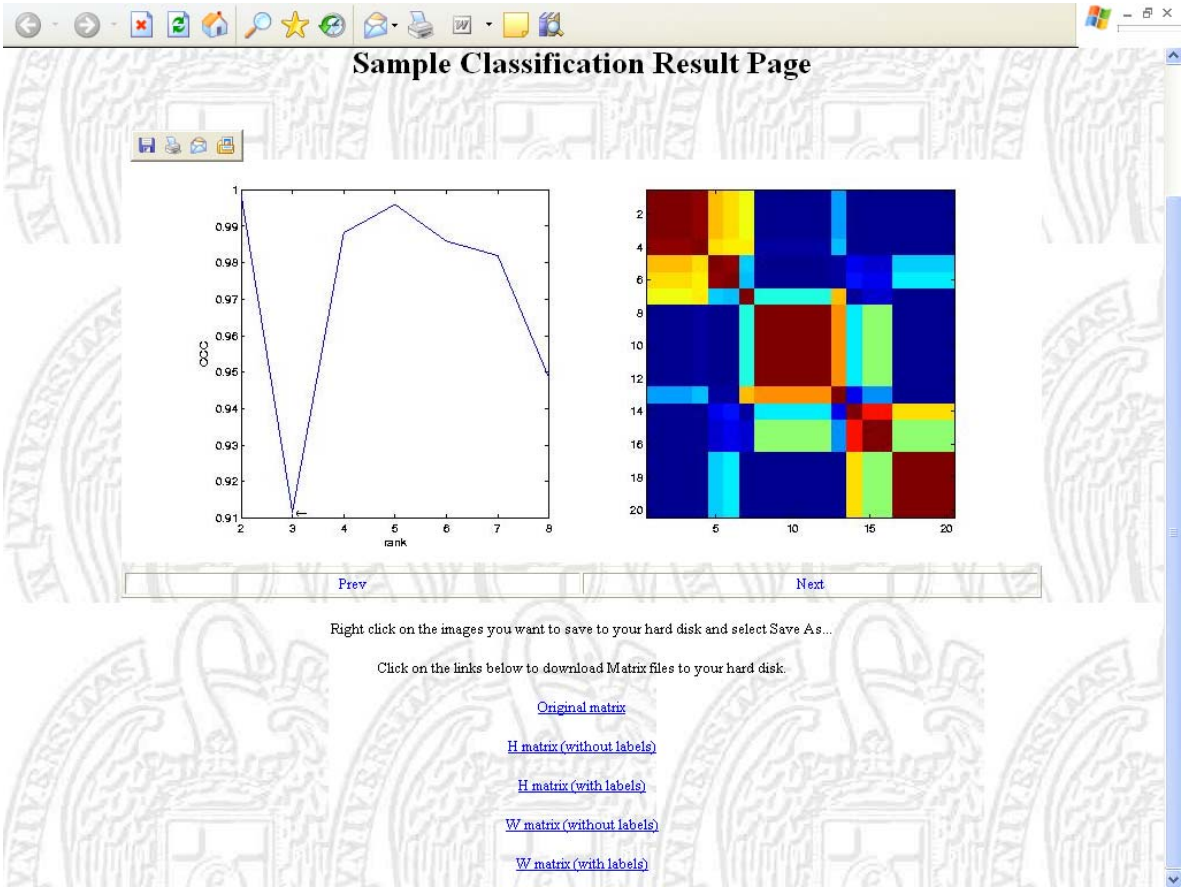

Figura 21: Resultado para Sample Clasification (II)

CONCLUSIONES Y TRABAJOS FUTUROS

CONCLUSIONES

Podemos concluir que los objetivos iniciales marcados para el proyecto se han llevado a cabo; se ha desarrollado un sistema web sencillo que cubre las necesidades del usuario potencial (investigador bioinformático), ofreciendo los servicios que ya ofrecía BioNMF y en la forma en la que estaban acostumbrados los investigadores, pero con las mejoras marcadas en los objetivos del desarrollo.

Una vez completado el proyecto y después de utilizarlo se pueden comprobar las ventajas del mismo y que entre otras han sido la que nos han llevado al desarrollo del proyecto, como pueden ser el acceso a la aplicación con sólo disponer un acceso a Internet, sin tener que instalar ninguna aplicación e independiente de la plataforma usada, y la mejora en el procesamiento del análisis y por tanto en la obtención del resultado, gracias al procesamiento en paralelo de los cálculos necesarios.

La mayor dificultad que nos surgió fue cómo afrontar las notificaciones entre los distintos procesos que intervienen en nuestra aplicación, como puede ser entre Matlab y la web, por ejemplo. Finalmente decidimos solucionarlo por medio de la máquina de estados y el script *'scanner'* que realiza las transiciones entre los distintos estados.

También nos surgieron algunos problemas con la visualización de las imágenes resultado, dado que esa parte se realiza en Matlab y no teníamos un conocimiento profundo de la materia. Gracias a la ayuda del profesor director y su equipo se consiguió que las distintas imágenes se mostraran correctamente.

TRABAJOS FUTUROS

Una posible e interesante mejora para el sistema web sería su integración con implementaciones eficientes del algoritmo NMF tanto en arquitecturas de altas prestaciones como en clusters y computación GRID [18], ya que la herramienta está diseñada para que funcione como un conector que pueda conectarse y desconectarse sin problemas a sistemas con arquitecturas de este tipo.

Se han realizado ya algunos estudios para su integración en modelos basados en flujos y más concretamente su implementación para la ejecución en una unidad de procesamiento gráfico (GPU) [18].

Los modelos de procesamiento basados en flujos permiten expresar el paralelismo inherente a una aplicación desacoplando el cómputo de los accesos a memoria. Bajo estos modelos, los datos se agrupan en flujos (*streams*) para ser procesados por *kernels* que generan nuevos *streams* de salida. En algunos dominios, dadas sus características, la implementación de las aplicaciones en arquitecturas de flujos es inmediata, pero en otros se hace necesario aplicar varias transformaciones para poder expresar las aplicaciones por medio de secuencias de *kernels* que operan sobre *streams*.

Las unidades de procesamiento gráfico de consumo (GPU) son uno de los tipos de arquitecturas en el que el elemento central de su diseño son las unidades funcionales y no las estructuras de control y almacenamiento típicas de los procesadores superescalares. Debido a su diseño alcanzan rendimientos pico elevados, por supuestos bastante superiores a los alcanzados por los procesadores de propósito general.

APÉNDICES

CÓDIGO SCRIPT MATLAB

```
function trataprefs
verbose = 0;
%reading preferences file
fid=fopen('Preferences.txt','rt');

%reading Transpose value
strlinea = fgetl(fid);
transpose = GetVarValue(strlinea);
if verbose fprintf(1,'transpose %s\n',transpose); end

%reading Normalization value
%possible norm values are: nonorm, subtract, scale, meanRows,
% meanColumns, subtractRows, subtractColumns, subtractBoth
strlinea = fgetl(fid);
norm = GetVarValue(strlinea);
if verbose fprintf(1,'normalization %s\n',norm); end

%reading Make positive value
%possible positive values are: nothing, expo, columns, rows, substract
strlinea = fgetl(fid);
positive = GetVarValue(strlinea);
if verbose fprintf(1,'positive %s\n',positive); end

%reading Method value
strlinea = fgetl(fid);
method = GetVarValue(strlinea);
if verbose fprintf(1,'method %s\n',method); end

%Reading Sample Classification vars
if (strcmp(method,'rsSample'))

    strlinea = fgetl(fid);
    minF = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'MinFactor %d\n',minF); end

    strlinea = fgetl(fid);
    maxF = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'MaxFactor %d\n',maxF); end

    strlinea = fgetl(fid);
    runs = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'Num Runs %d\n',runs); end

%Reading Biclustering vars
elseif (strcmp(method,'rsBiclustering'))
    %Reading Factores
    strlinea = fgetl(fid);
    ba_Factors = str2num(GetVarValue(strlinea));
```

```
    if verbose fprintf(1,'Factores %d\n',ba_Factors); end
    %Reading Sparseness
    strlinea = fgetl(fid);
    ba_Sparseness = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'Sparseness %f\n',ba_Sparseness); end
    %Reading Iterations
    strlinea = fgetl(fid);
    ba_Iterations = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'Iterations %d\n',ba_Iterations); end
    %Reading Stopping
    strlinea = fgetl(fid);
    ba_Stopping = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'Stopping %d\n',ba_Stopping); end
    %Reading Runs
    strlinea = fgetl(fid);
    ba_Runs = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'Runs %d\n',ba_Runs); end

%Reading Standard NMF vars
elseif (strcmp(method,'rsStandard'))
    %Reading Factores
    strlinea = fgetl(fid);
    st_Factors = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'Factores %d\n',st_Factors); end
    %Reading Iterations
    strlinea = fgetl(fid);
    st_Iterations = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'Iterations %d\n',st_Iterations); end
    %Reading Stopping
    strlinea = fgetl(fid);
    st_Stopping = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'Stopping %d\n',st_Stopping); end
    %Reading Runs
    strlinea = fgetl(fid);
    st_Runs = str2num(GetVarValue(strlinea));
    if verbose fprintf(1,'Runs %d\n',st_Runs); end
    %Reading Combine
    strlinea = fgetl(fid);
    st_chkCombine = GetVarValue(strlinea);
    if verbose fprintf(1,'Combine %d\n',st_chkCombine); end
end
fclose(fid);
%end of reading preferences file

%reading matrix without_labels
withoutM = load('MatrixMat.txt');
%Normalization
Vnorm = Normalization(withoutM, norm);
%We make it positive. We enter Vnorm Matrix
VPos = Positive(Vnorm, positive);
%Separate matrix

if (strcmp(method,'rsStandard'))
    [VW, VH] = nmf(VPos,st_Factors,verbose);
elseif (strcmp(method,'rsBiclustering'))
    [VW, VH] = biclustering(VPos,ba_Factors,ba_Runs,ba_Sparseness);
```

```
elseif (strcmp(method,'rsSample'))
    [VW, VH] = nmf(VPos,minF,verbose);
    consensus = nmfconsensus(VPos,minF,maxF,runs,verbose);
    [ordcons,clustid,ordindex,coph] =
nmforderconsensus(consensus,minF,maxF);

    % How to use printCCC to print all factors
    [kmax,m,m]=size(ordcons);
    for i=minF:maxF
        u=reshape(ordcons(i,:,:),m,m);
        printCCC(coph,u,i,['image-',int2str(i-minF+1)]);
    end
end

%Saving VW and VH to disk
save('MatrixW.txt','VW','-ascii','-tabs');
save('MatrixH.txt','VH','-ascii','-tabs');
SaveLabels;

%Change state file to finished
fid = fopen('State.txt','w');
fprintf(fid,'finished');
fclose(fid);
%get out from matlab
exit;


function SaveLabels
%Creating labels VH VW files
    fWLBL = fopen('MatrixWLBL.txt','w');
    fprintf(fWLBL,'No labels on original matrix');
    fclose(fWLBL);
    fHLBL = fopen('MatrixHLBL.txt','w');
    fprintf(fHLBL,'No labels on original matrix');
    fclose(fHLBL);

    fid = fopen('MatrixCab.txt','r');
    if (fid >= 0) %If fid = -1. It doesn't exists
        fHLBL = fopen('MatrixHLBL.txt','w');
        fprintf(fHLBL,'%s\n',fgetl(fid));
        fHid=fopen('MatrixH.txt','r');
        while feof(fHid) == 0
            fprintf(fHLBL,'%s\n',fgetl(fHid));
        end
        fclose(fHid);
        fclose(fHLBL);

        fWLBL = fopen('MatrixWLBL.txt','w');
        fWid=fopen('MatrixW.txt','r');
        while feof(fWid) == 0
            fprintf(fWLBL,'%s ',fgetl(fid));
            fprintf(fWLBL,'%s\n',fgetl(fWid));
        end
        fclose(fWid);
        fclose(fWLBL);
```

```
        fclose(fid);
    end
%Function to norm matrix V
function VNorm = Normalization(V, normmethod)
        %First calculate global matrix mean
        SizeData = size(V);
        numColumns = size(V, 2);
        numRows = size(V, 1);

    switch lower(normmethod)
        case 'nonorm'
            VNorm = V;
        case 'subtract'
            %calculate global mean
               Vec = reshape(V, [prod(SizeData), 1]);
               VMean = mean(Vec);
               VNorm = V - VMean;
        case 'scale'
                        %First divide matrix mean by columns
                        e = ones(1,numColumns);
            VMeanCol = mean(V')';
               VDiv = V ./ (VMeanCol*e);
               %then normalize rows
               VNorm = normr(VDiv);
        case 'meanrows'
            %First calculate 0 mean rows matrix
                        f = ones(numRows,1);
            VMeanRow = mean(V);
            VZeroMeanRow = V - (f*VMeanRow);
            % now calculate stardard deviation and divide
            s = std(V);
            VNorm = VZeroMeanRow ./ (f*s);
        case 'meancolumns'
                        %First calculate 0 mean cols matrix
                        e = ones(1,numColumns);
            VMeanCol = mean(V')';
               VZeroMeanCol = V - (VMeanCol*e);
            % now calculate standard deviation and divide
            s = std(V')';
            VNorm = VZeroMeanCol ./ (s*e);
        case 'subtractrows'
            %Calculate  mean row from matrix
                        f = ones(numRows,1);
            VMeanRow = mean(V);
            VNorm = V - (f*VMeanRow);
        case 'subtractcolumns'
                        %Subtract matrix mean by columns
                        e = ones(1,numColumns);
            VMeanCol = mean(V')';
               VNorm = V - (VMeanCol*e);
        case 'subtractboth'
            %First rest Row mean matrix
                        f = ones(numRows,1);
            VMeanRow = mean(V);
            VRestRow = V - (f*VMeanRow);
            %Second rest Column mean of rest row matrix result
```

```
                    e = ones(1,numColumns);
            VMeanCol = mean(VRestRow')';
                VNorm = VRestRow - (VMeanCol*e);
        otherwise
            VNorm = V;
        end


%Fuction to make postive V matrix
function VPos = Positive(V, posmethod)
        themin = min(min(V));
        themax = max(max(V));

        if abs(themin) < abs(themax)
        absmax = abs(themax);
        else
        absmax = abs(themin);
        end

    switch lower(posmethod)
        case 'nothing'
            VPos = V;
        case 'expo'
                    VPos = exp(V *  (3.0/absmax));
        case 'substract'
            VPos = V - themin;
        otherwise
            VPos = V;
        end


% hand made strtrim
function strt = miStrTrim(stringtotrim)
    strt = fliplr(deblank(fliplr(deblank(stringtotrim))));


%function that reads the value from a string after the ":" separator
function nameVar = GetVarValue(tempstr)
    tempstr = miStrTrim(tempstr);
    nameVar = miStrTrim(tempstr(find(tempstr == ':')+1:length(tempstr)));


function  printCCC(coph, ordcons, k, name)
%
% Pedro Carmona-Saez
% BioComputing Unit
% pcarmona@cnb.uam.es

% Call: printCCC(coph,ordcons,'name')
% coph: 1d array containing cophenectic correlation coefficients(output
of nmforderconsensus function)
% ordcons: 2d matrix
% name: Name of output file

[r,c]=size(coph);
f=figure;
```

```
set(gcf, 'Position', [250 500 800 350]);

subplot(1,2,1);
plot(2:c, coph(2:c));

text(k,coph(k),' \leftarrow','FontSize',10);
xlabel('rank','FontSize',10);
ylabel('CCC','FontSize',10);
set(gca,'XTick',2:1:c);
xlim([2 c]);
subplot(1,2,2);
imagesc(ordcons);

set(gcf, 'PaperPositionMode', 'auto');
print ('-djpeg',name);
close(f);
```

CÓDIGO FICHERO SCANNER

```
#!/bin/bash

cd ..
while [ 1 ]; do
for i in *
do
        if [ -d $i ]
        then
                # We enter if it is a directory
                cd $i
                # Looking for State.txt file
                if [ -f State.txt ]
                then
                        grep -q prepared State.txt
                        # If state = prepared -> send to matlab
                        if ( test $? -eq 0 )
                        then
                                nohup matlab -r trataprefs -nodesktop -
nosplash -nodisplay -nojvm &
                                echo Sent to Matlab > State.txt
                        fi

                        grep -q finished State.txt
                        # If state = finished -> Email and copy Result
page

                        if ( test $? -eq 0 )
                        then
                                let numImages=0
                                grep -q rsSample Preferences.txt
                                if ( test $? -eq 0 )
                                then
                                        rm -f result.jsp
                                        cp ../resultSample.jsp result.jsp
```

```
                                        numMaxImages=`grep 'Max Factors:'
Preferences.txt | awk '{print $3}'`
                                        numMinImages=`grep 'Min Factors:'
Preferences.txt | awk '{print $3}'`
                                        let numImages=$numMaxImages-
$numMinImages
                                fi
                                        grep -q rsStandard Preferences.txt
                                        if( test $? -eq 0 )
                                        then
                                                rm -f result.jsp
                                                cp ../resultStandard.jsp
result.jsp

                                        else
                                                grep -q rsBiclustering
Preferences.txt
                                                if ( test $? -eq 0 )
                                                then
                                                        rm -f result.jsp
                                                        cp
../resultBiclustering.jsp result.jsp
                                                        numTotImages=`grep
'Factors:' Preferences.txt | awk '{print $2}'`
                                                        let
numImages=$numTotImages
                                                fi
                                        fi
                                fi
                                txtImage=`echo var numImages =
$numImages\;`
                                # Cambiamos el numero de imagenes
dependiendo de las preferencias
                                `sed "s/var numImages = 0;/$txtImage/"
result.jsp > tmpfile && mv tmpfile result.jsp`
                                # A□□mos el directorio de la jsp para q
coja las imagenes image y cof
                                `sed "s/#DIR#/$i\//" result.jsp > tmpfile
&& mv tmpfile result.jsp`
                                grep -q Email Preferences.txt
                                if ( test $? -eq 0 )
                                then
                                        # Mandamos correo
                                        email=`grep 'Email:'
Preferences.txt | awk '{print $2}'`
                                        mutt -s "bioNMF: Job Finished" -e
"my_hdr From: bionmf <bionmf@bionmf.com>" $email < Instructions.txt
                                        `sed "s/Email:/E-Mail:/"
Preferences.txt > tmpfile2 && mv tmpfile2 Preferences.txt`
                                        echo Email sent to $email
                                        # Cabiamos estado a terminado
                                        echo Terminated. Email sent >
State.txt
                                else
                                        # Cabiamos estado a enviado
```

```
                                        echo Terminated. No Email >
State.txt
                                fi
                                if [ -f enddate.txt ]
                                then
                                        echo enddate already created
                                else
                                        # Fecha en segundos
                                        date +%s > enddate.txt
                                fi
                        else
                                echo Old state
                                cat State.txt
                        fi
                fi
                # Si existe una fecha de fin miramos si ha pasado un mes
para borrar el directorio
                if [ -f enddate.txt ]
                then
                        timenow=`date +%s`
                        timethen=`cat enddate.txt`
                        let "timediff = $timenow - $timethen"
                        #2592000 son los segundos que tiene un mes.
Miramos la diferencia
                        if [ "$timediff" -ge 2592000 ]
                        then
                                echo Deleting directory
                                cd ..
                                rm -rf $i
                        else
                                cd ..
                        fi
                else
                        cd ..
                fi
        fi
sleep 2s
done
sleep 1m
done
exit 0
```

CÓDIGO PROCESO DEMONIO

```
#!/bin/bash
case "$1" in
        start)
       echo "Starting scanner.."
       buscaPid=`ps -C scanner.sh -o pid=`
       if [ "$buscaPid" = "" ]
         then
            nohup ./scanner.sh > /dev/null &
            echo "Scanner is up"
```

```
          else
             echo "Scanner is already up. Stop it first"
          fi
          ;;
        stop)
           echo "Stopping scanner.."
           miPid=`ps -C scanner.sh -o pid=`
           kill -9 $miPid
         echo "scanner stopped"
         ;;
        *)
          echo "Usage: $0 {start|stop}"
          exit 1
          ;;
esac
exit 1
```

BIBLIOGRAFÍA

[1] Lee DD, Seung HS: **Learning the parts of objects by non-negative matrix factorization.**
Nature 1999, 401:788-91

[2] Carmona-Saez P, Pascual-Marqui RD, Tirado F, Carazo JM, Pascual-Montano A: **Biclustering of gene expression data by non-smooth non-negative matrix factorization.**
BMC Bioinformatics 2006, 7:78.

[3] Heger A, Holm L: **Sensitive pattern discovery with 'fuzzy' alignments of distantly related proteins.**
Bioinformatics 2003, 19(Suppl 1):i130-7.

[4] Pehkonen P, Wong G, Toronen P: **Theme discovery from gene lists for identification and viewing of multiple functional groups.**
BMC Bioinformatics 2005, 6:162.

[5] Chagoyen M, Carmona-Saez P, Shatkay H, Carazo JM, Pascual-Montano A: **Discovering semantic features in the literature: a foundation for building functional associations.**
BMC Bioinformatics 2006, 7:41.

[6] Wall ME, Dyck PA, Brettin TS**: SVDMAN – singular value decomposition analysis of microarray data.**
Bioinformatics 2001, 17:566-8.

[7] Lee SI, Batzoglou S: **Application of independent component analysis to microarrays.**
Genome Biol 2003, 4:R76.

[8] Dai JJ, Lieu L, Rocke D: **Dimension reduction for classification with gene expression microarray data.**
Stat Appl Genet Mol Biol 2006, 5:Article6.

[9] Pascual-Montano A, Carazo JM, Kochi K, Lehmann D, Pascual-Marqui RD. **Non-smooth Non-Negative Matrix Factorization (bioNMF)** *IEEE Trans. on Pattern Analysis and Machine Intelligence* 2006 28(3):403-415.

[10] bioNMF: A tool for non-negative matrix factorization in biology http://www.dacya.ucm.es/apascual/bioNMF/

[11] Portal de programación en castellano
Tutoriales sobre Java, J2EE, JSPs y servlets, etc
http://www.programacion.com/

[12] Site de Java Sun
J2EE, J2SE, Tutoriales, Foros, Ejemplos de Código Fuente, etc.
http://java.sun.com/

[13] Diversos portales consultados sobre java.
http://www.javasoft.com/
http://www.javaworld.com/
http://www.javahispano.com/

[14] **Struts in Action. Building web applications with the leading Java framework.**Ted Husted. Manning Publications Co. 2003.

[15] Servidor de Aplicaciones Tomcat
Portal de desarrollo de Tomcat en el Apache Jakarta Project.
http://jakarta.apache.org/tomcat

[16] Java en Castellano. Servlets y JSP
http://java.programacion.net/servlets_jsp/index.php

[17] Consulta de información variada.
http://es.wikipedia.org

[18] E. Mejía R, J.I. Gómez, M. Prieto, A. Pascual-Montano, F. Tirado. **Programación bajo un modelo basado en flujos. La factorización NMF como caso de estudio**. *ArTeCS Group,Universidad Complutense de Madrid.*

[19] Brunet JP, Tamayo P, Golub TR, Mesirov JP**: Metagenes and molecular pattern discovery using matrix factorization.** Proc Natl Acad Sci USA 2004, 101:4164-4169.